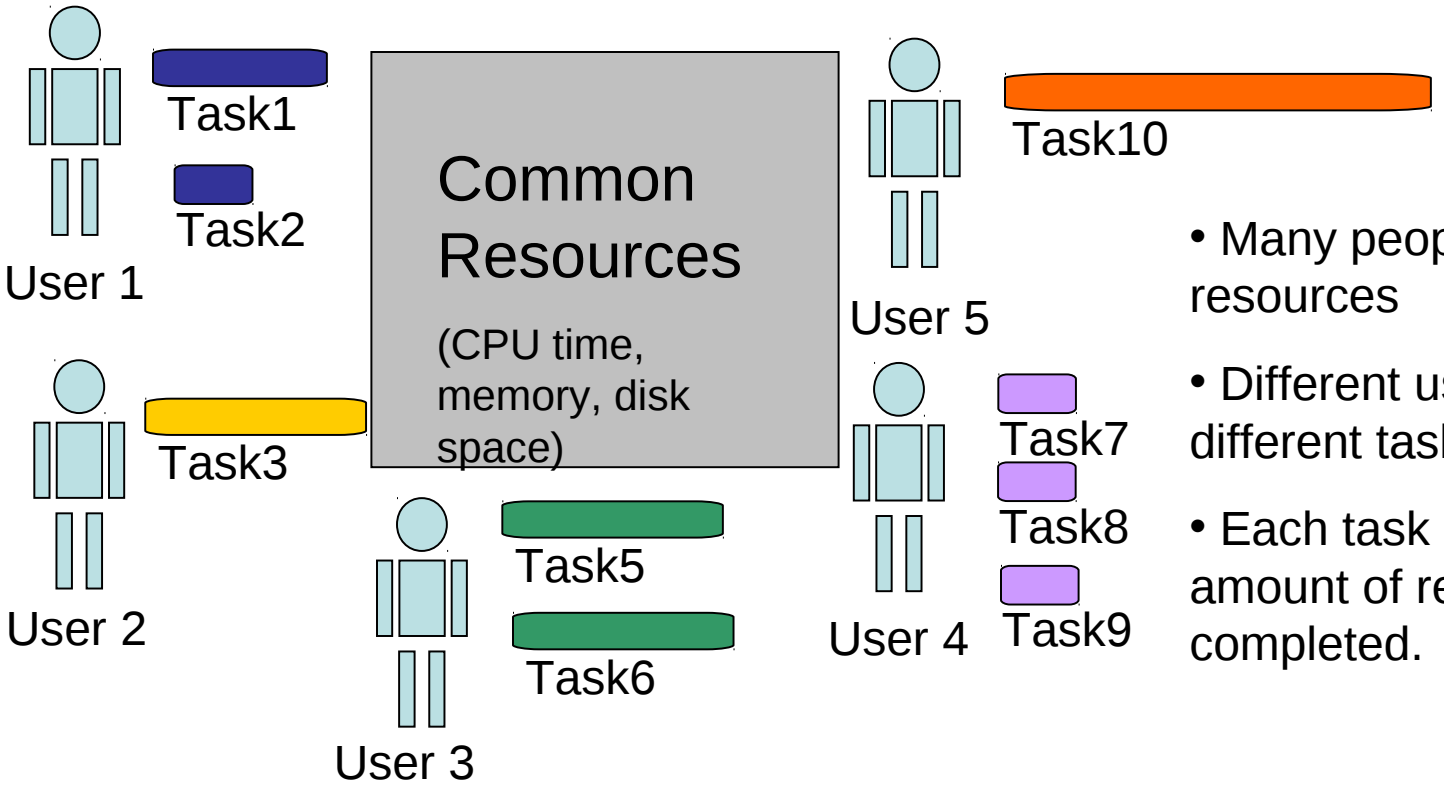
# Resource Allocation Policies

Volodymyr Sergiievskyi

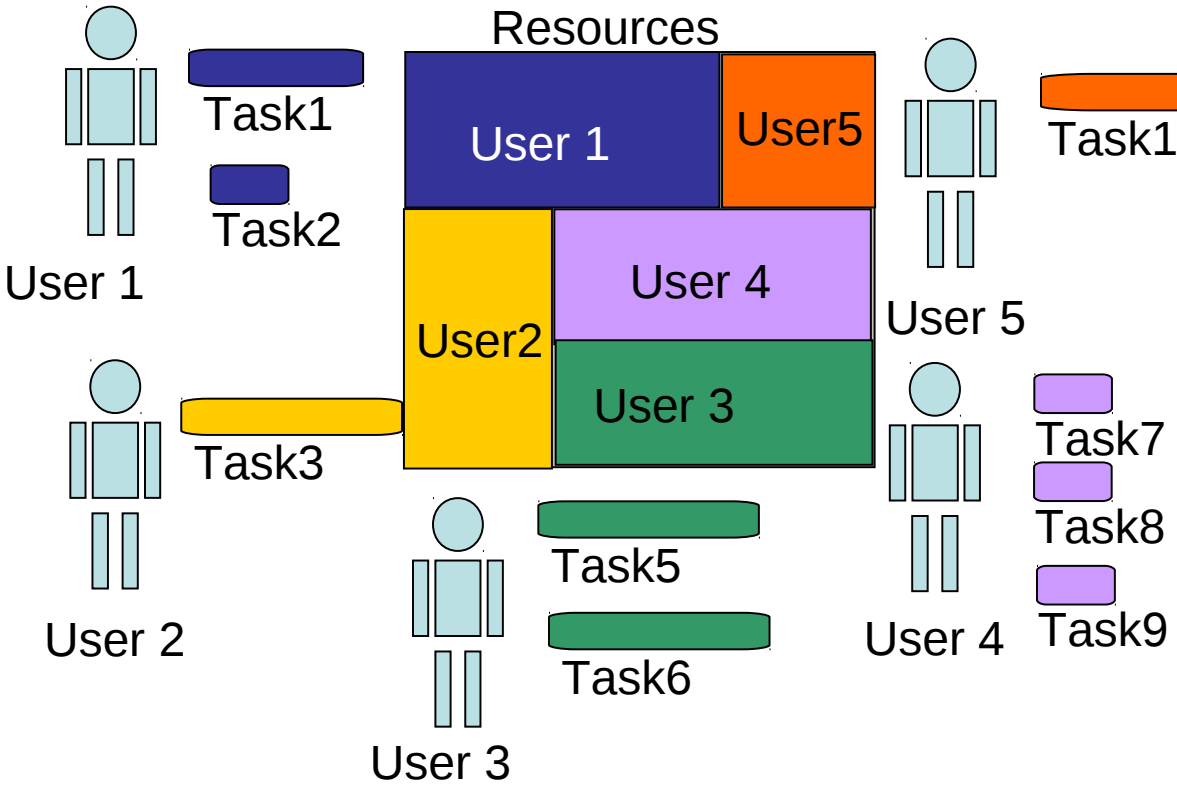
# Resource Allocation Policy




- Many people use the same resources
- Different users have different tasks
- Each task need some amount of resources to be completed.

Resource allocation policy determines which resources will be allocated to each user in each moment of time

# Static Resource Allocation




Static allocation:

Each user have access to some part of common resources.

**Advantage:** no conflicts

**Disadvantages**:

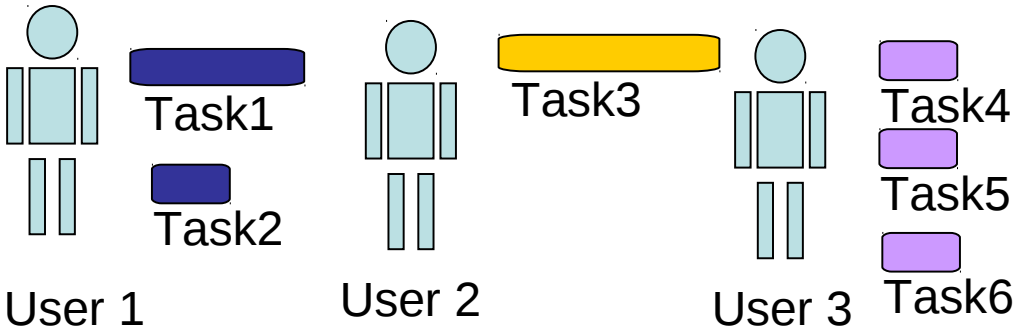
-small amount of resources for each user

-no possibility to run big parallel tasks

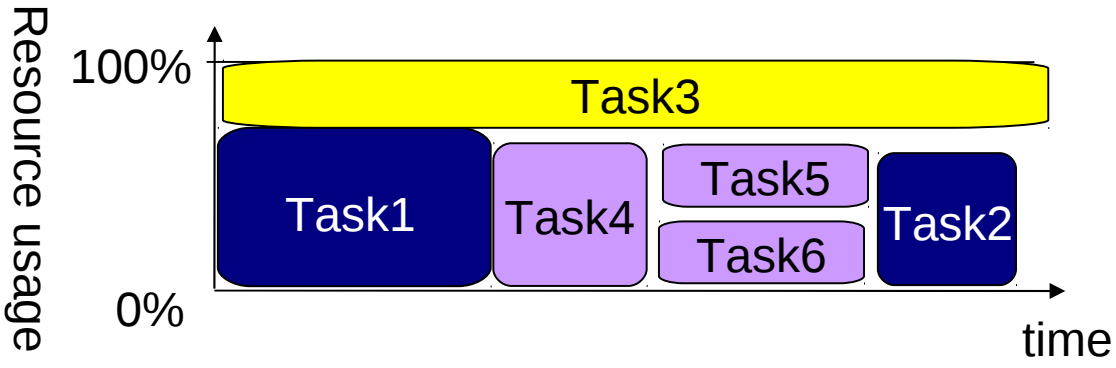
-the most of time supercomputer is idle

Static resource allocation is not good for large computational complexes with many CPU

# Dynamic Resource Allocation

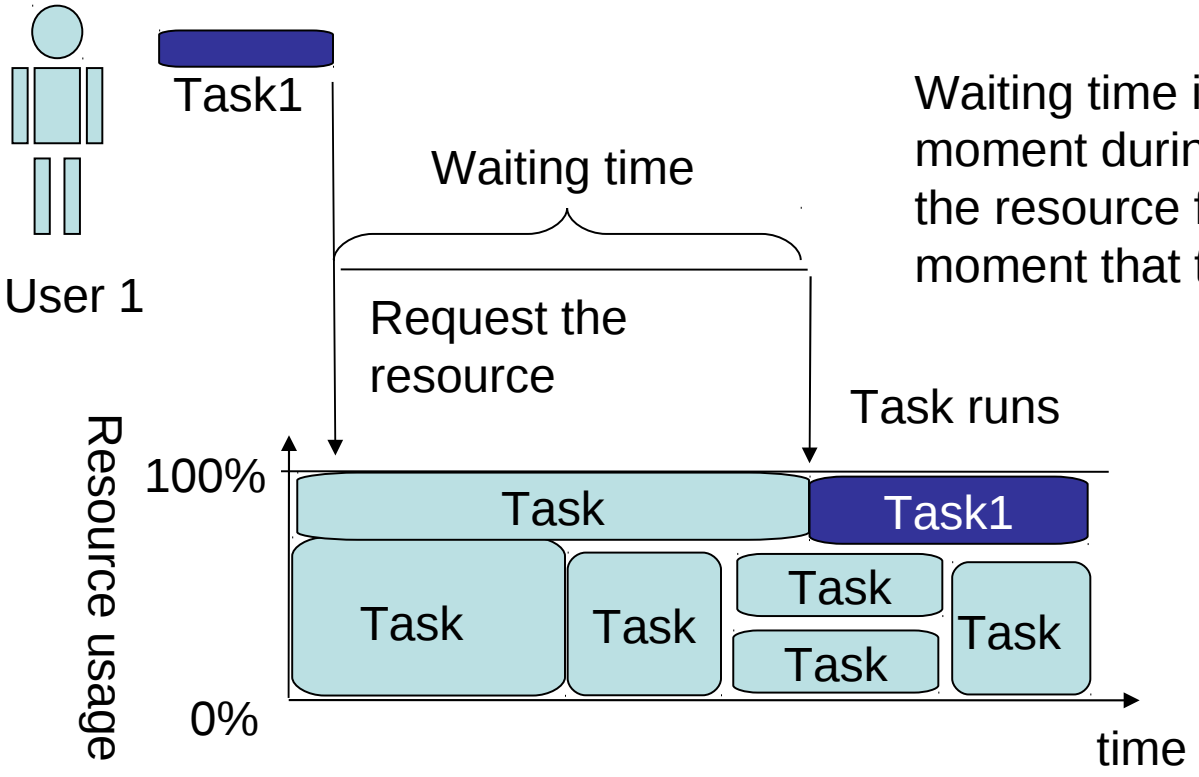



In dynamic resource allocation resources are not connected to the users, but are allocated by the users' request.




Dynamic resource allocation allows to use the resources more optimal and to increase an average load of the supercomputer.

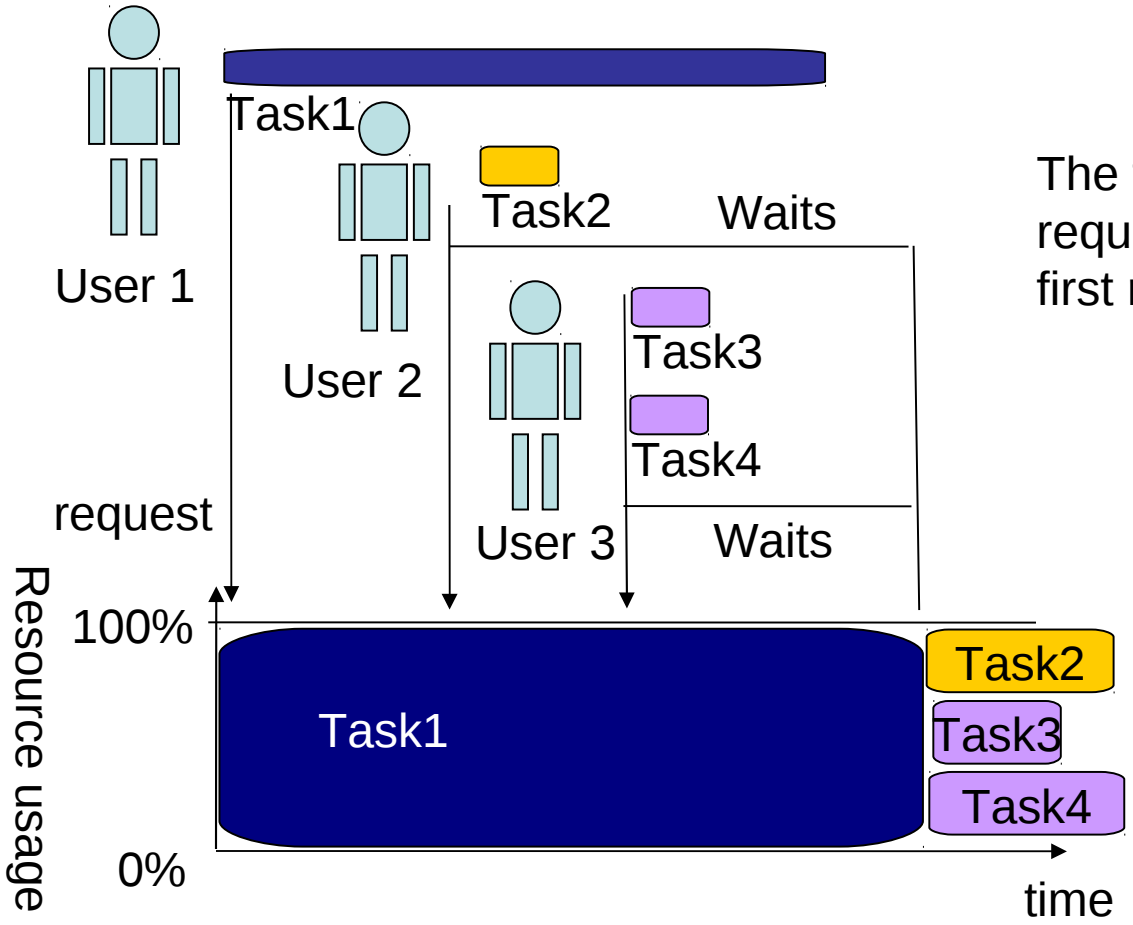
# Waiting Time




Waiting time is the time between the moment during that an user requests the resource for his task and the moment that the task actually runs

Waiting time and an average system load are the main criteria to an judge effectiveness of the resource allocation policy.

# “First Come First Served” Policy



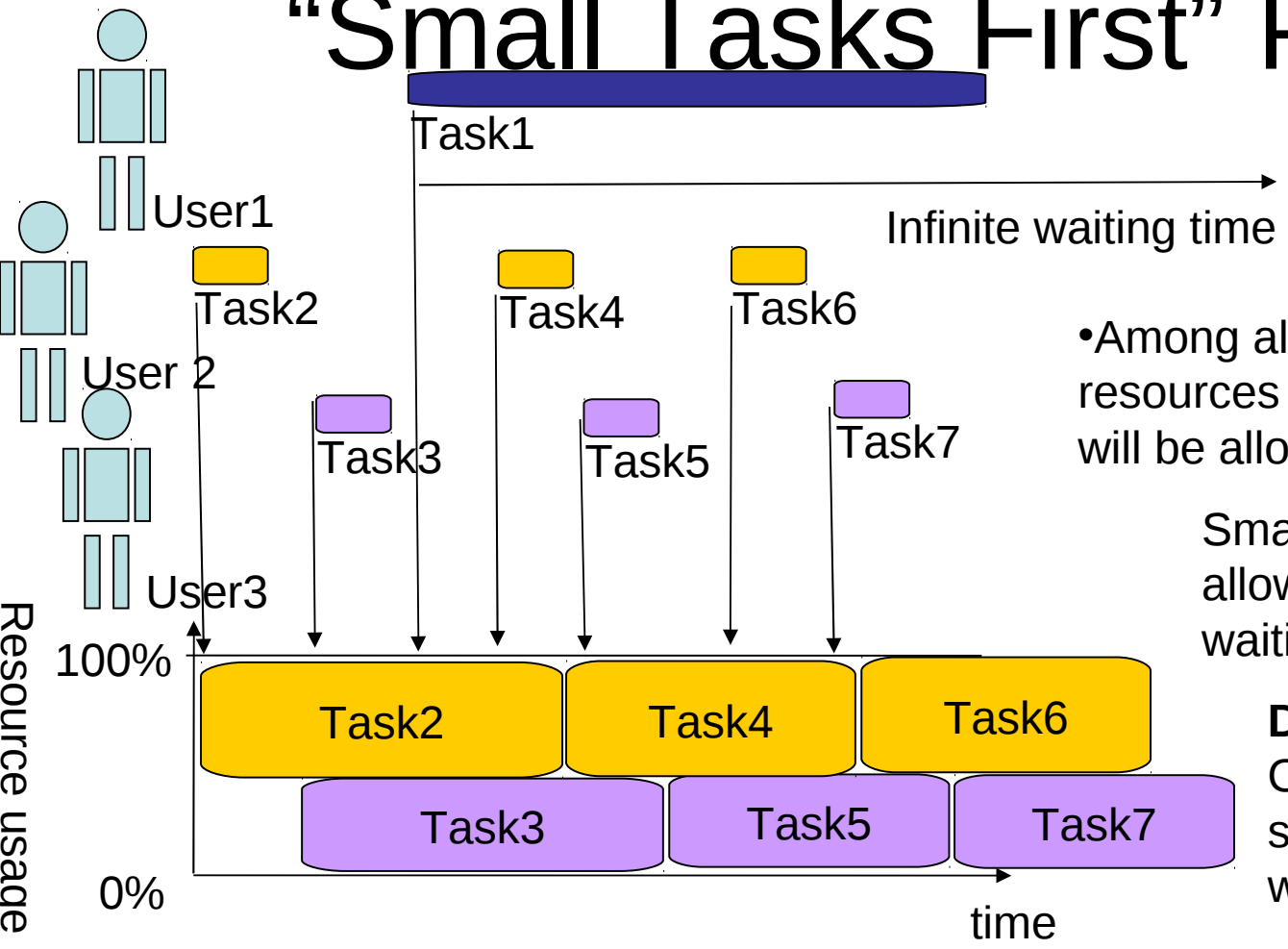

The task which requests the resource first runs the first

**Disadvantage:**

**I**f one user submits a big task others should wait until this task ends, even if their own tasks are small

First come first served policy works good only if users run only relatively small tasks.

# “Small Tasks First” Policy



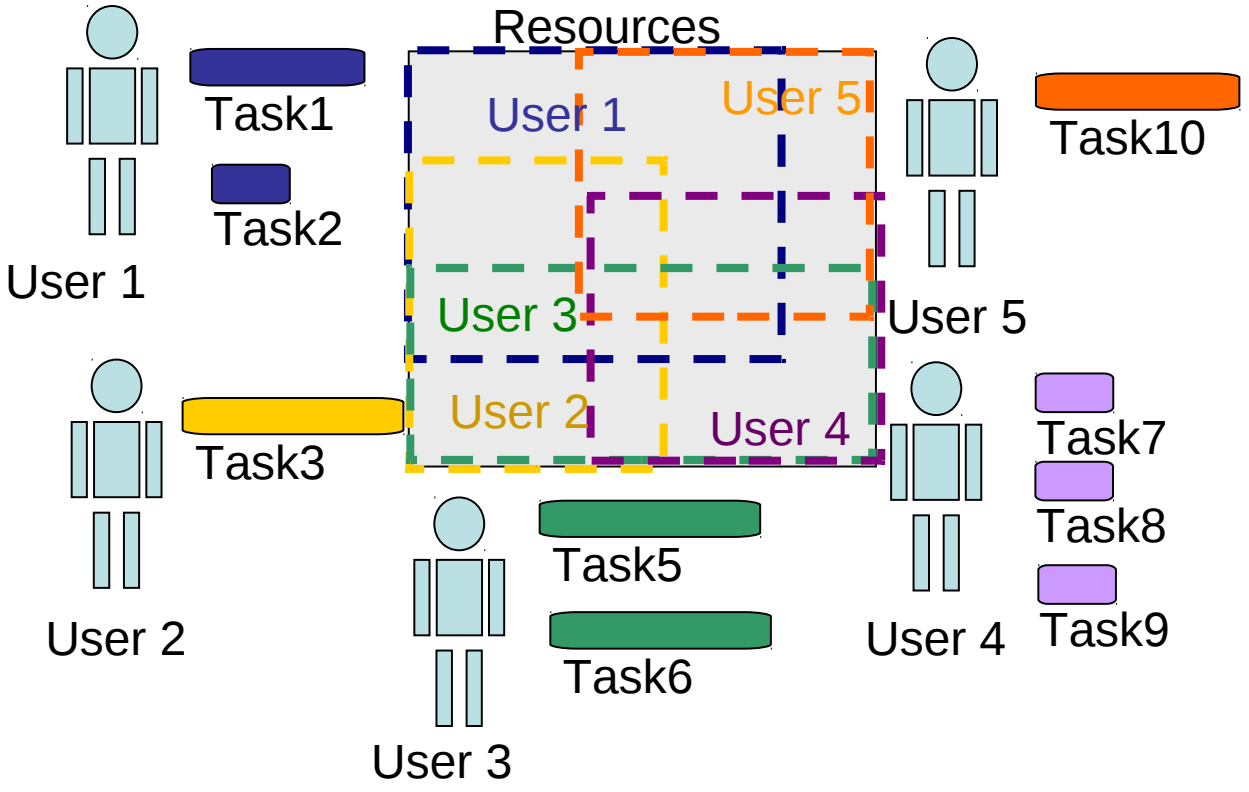

•Among all resource requests resources for the small tasks will be allocated the first

Small tasks first policy allows to decrease the waiting time for small jobs

**Disadvantage:**
One can imagine a situation when a large task will never start

“Small tasks first” policy should be used only if there is some additional check for infinite waiting time for large tasks

# Resource Limits



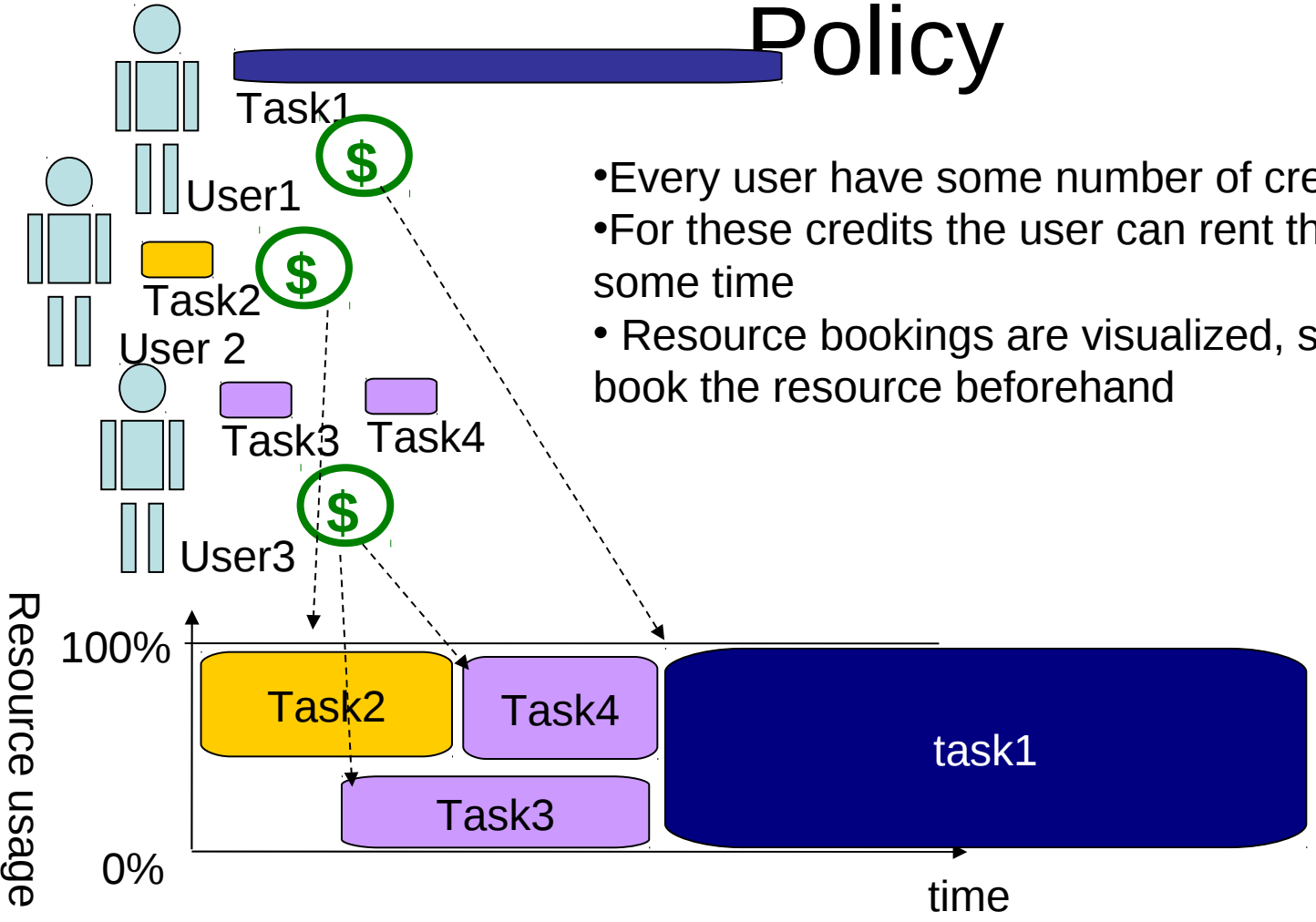

•Resource limits are introduced to avoid full occupation of the supercomputer by one user

**Advantage:** better waiting time and average load

**Disadvantage:** nobody can use the full power of the supercomputer

Resource limits are good for big supercomputers where no task will need it's full computational power

# “Credits” Resource Allocation Policy




- Every user have some number of credits
- For these credits the user can rent the resource for some time
- Resource bookings are visualized, so the user can book the resource beforehand

Rent system is more flexible than system of limits and allows to avoid many disadvantages of other resource allocation policies

# Simulation Setup

3500 CPU, 250 Users, 300 days

Three category of users:

**Group 1:** “Long tasks”: Task 40±20 days 10±5 CPU (33% of users)
**Group 2:** “Big short tasks”: Task 10±5 days, 200±100 CPU (33% of users)
**Group 3:** “Many small tasks”: Task 5±3 days, 10±5 CPU (33% of users)

Total Resources= 3500*300 CPU/days.

Each user submit tasks at random moments of time. The total number of CPU/days is the same for each user.

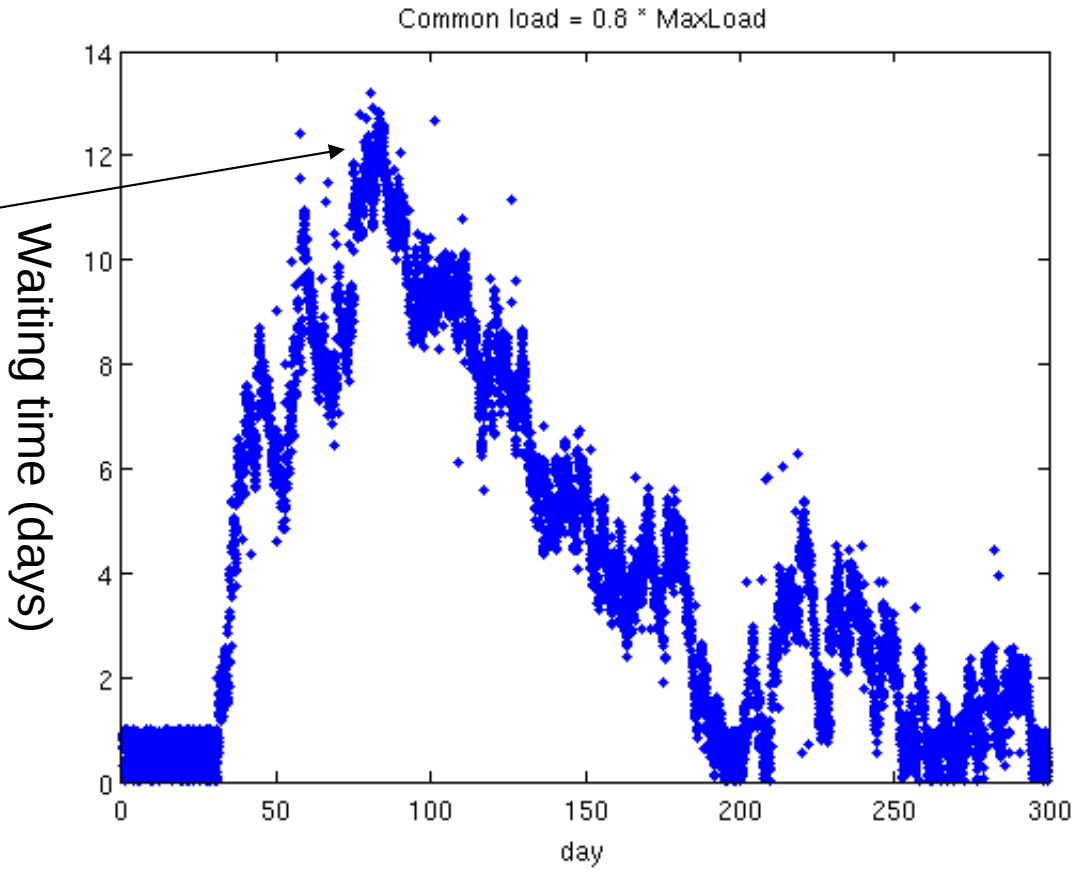
In total, users submit α*TotalResources CPU/days, where α is a load coefficient.

Resources are allocated according to the First come – first served policy

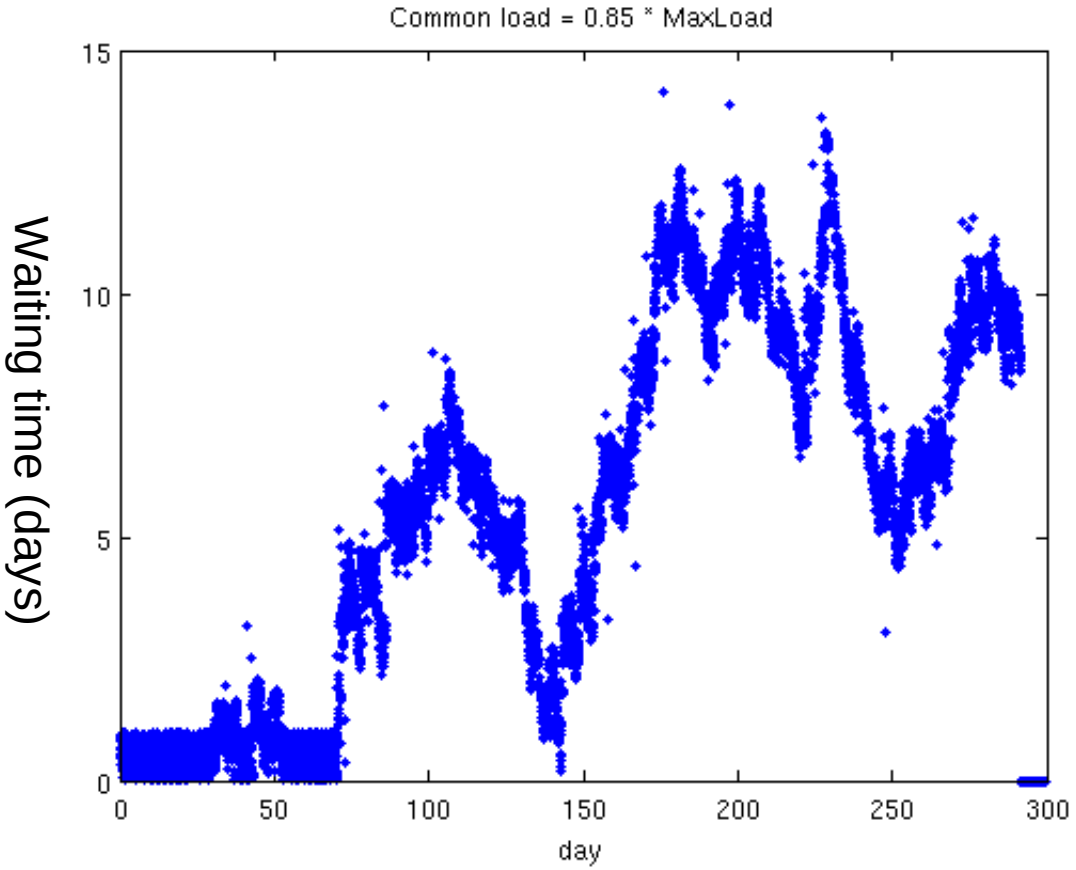
# Results

Load coefficient α = 0.8 Resources (users are submitting tasks, that are using in total 80% of the CPU resource per 300 day )

Waiting time peak



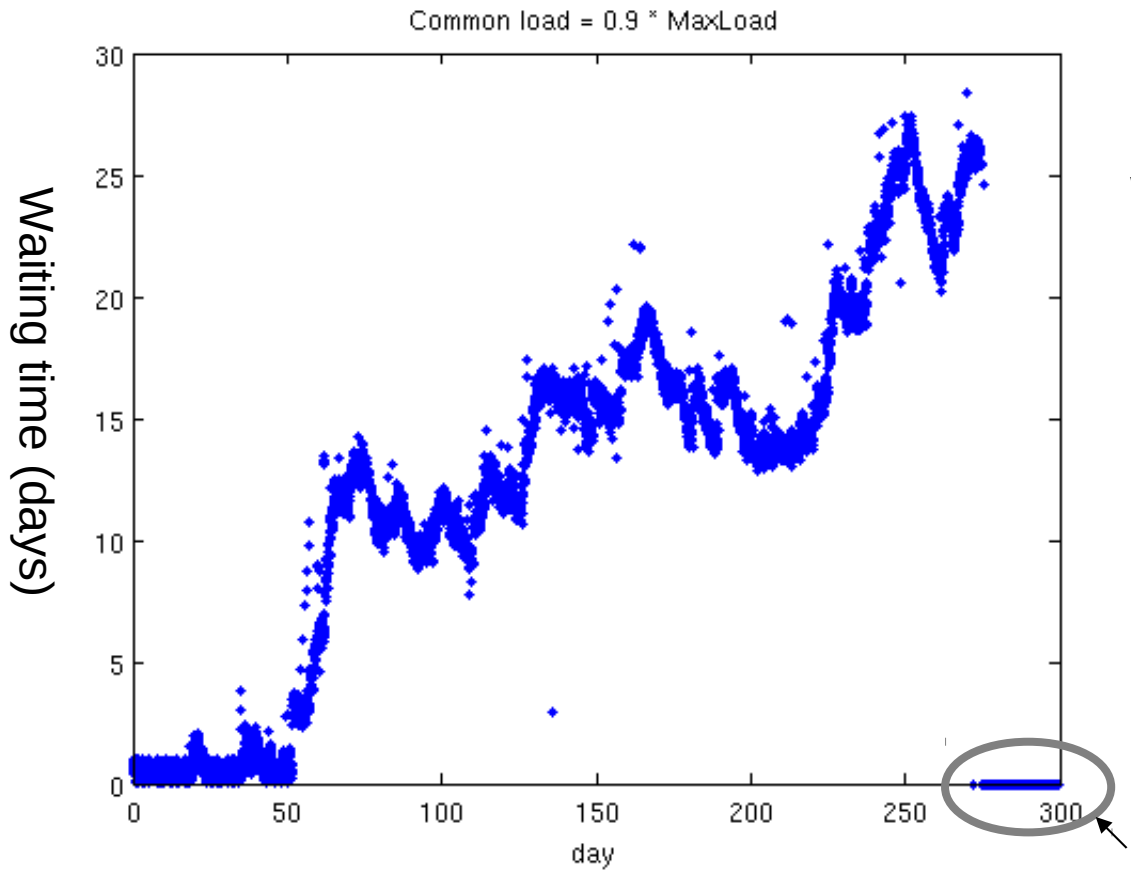
# α = 0.85




Waiting time grows, but then decays..

Seems, that α=0.85 is near the critical value.

# α = 0.9




Waiting time is growing constantly. α = 0.9 is over the critical load.

Tasks which did not managed to run

# Conclusion

- There is a critical value of load constant and it is about $\alpha = 0.85$ (thus if all the users send tasks which in total occupy more that 0.85 of all CPU/days, it will lead to uncontrolled growth of the waiting time)
- it is not clear how to force the users do not to send more tasks than the critical amount (which is also unknown beforehand)

# Cancellation Rate

- In the reality the waiting time cannot grow infinitely. Users will not wait forever, thus tasks will be canceled after some maximal waiting time.

- It is reasonable to suggest that maximal waiting time is proportional to the duration of the task:

  CancellationTime = β*DurationOfTask

  where β is a cancellation coefficient

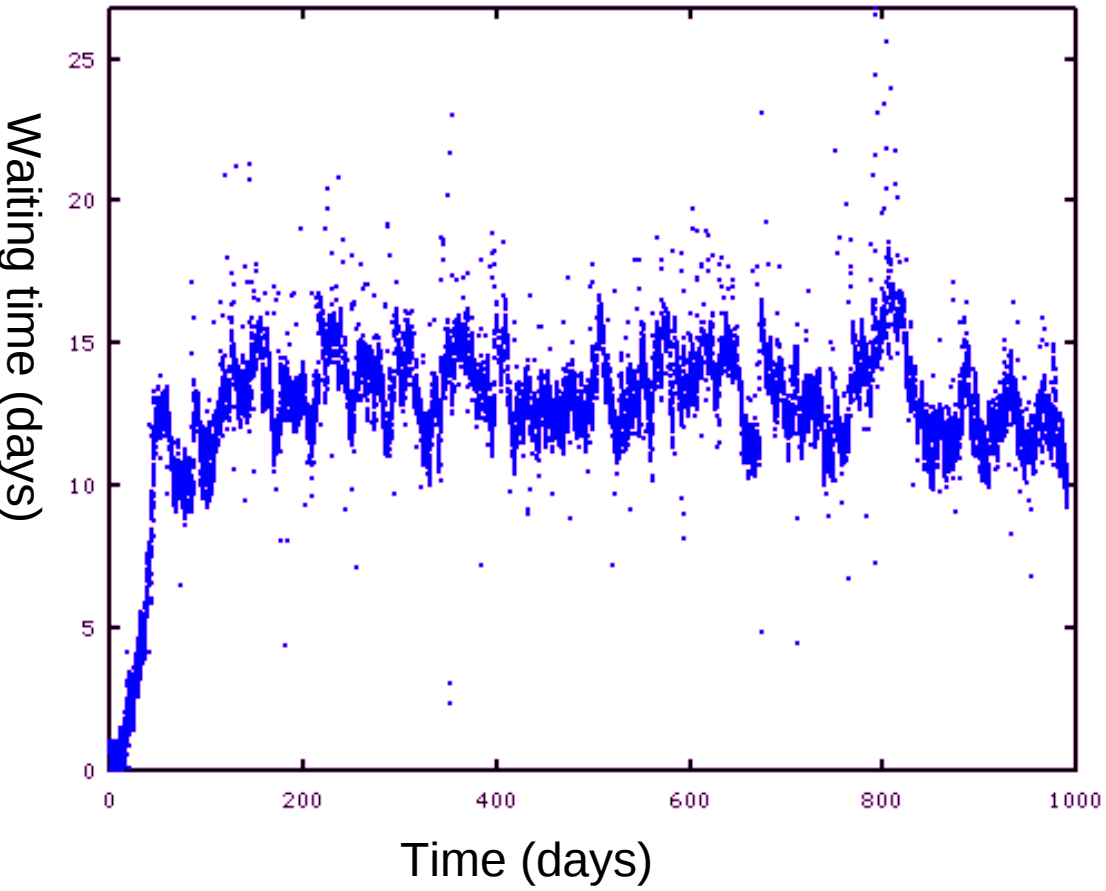
- User's *cancellation rate* shows the fraction of user's tasks which were cancelled

$$CancellationRate = N_{canceledTasks} / N_{tasks}$$

# Simulation Setup

- 3500 CPUs, 1000 days, 250 Users
- Three groups of users:
  - **Group 1**: “Long tasks” (33% of users)
  - **Group 2**: “Big short tasks” (33% of users)
  - **Group 3**: “Many small tasks” (33% of users)
- Load coefficient α=1.25, which includes:
  - 50% - active users ( load = 1*TotalResources)
  - 50% non-active users ( load = 0.25*TotalResources)
- Cancellation coefficient β=2

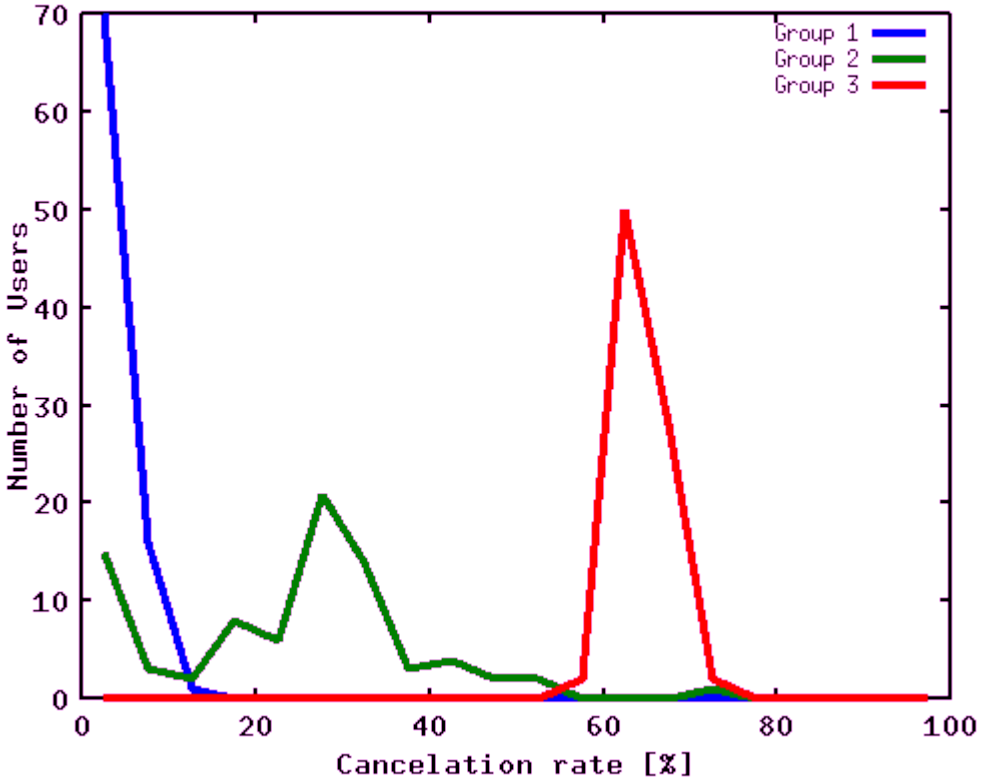
# Results for First Come First Served




- Average System load > 99%
- Waiting does not grow unlimited, and stabilizes after 50-100 days
- Average waiting time = 11.8 days
- Average cancellation rate =27%

If users will cancel their tasks the waiting time will stabilize after some period. However none of the parameters are perfect in such system.

# Cancellation Rate For Different User Groups




- Different groups of users have different mean cancellation rates
- A user in Group 3 who runs many small tasks have cancellation rate over 60% (more than every second task is cancelled)

First come first served policy can be discriminative for some user groups

# Limits System

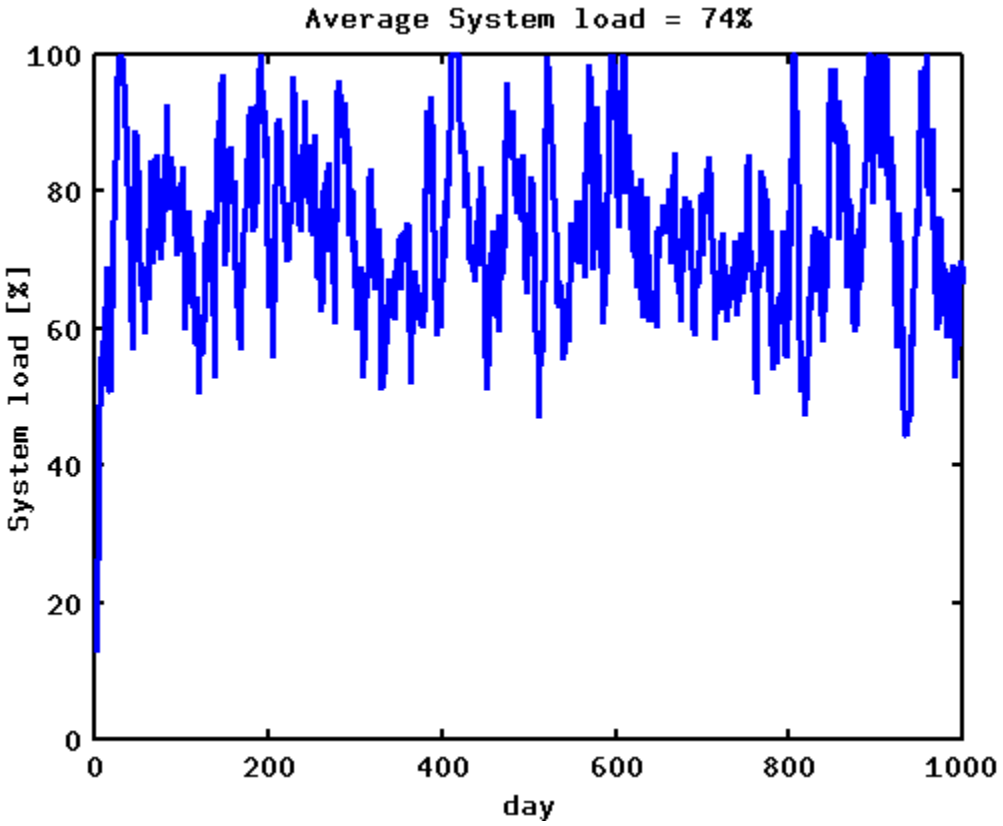
- In order to avoid an overload of the system and cancellations of the tasks one can introduce limits for users.

- Because α=0.85 is near the critical value it is reasonable to introduce the following limit of CPU load for each user:

Limit = 0.85 / NumerOfUsers * TotalResources

- As before we have 3 groups of users--> in each group
  - 50% - active users (Use maximal possible amount of resources)
  - 50% non-active users (Total load = 0.25 * TotalResources)

# Results for Limits System



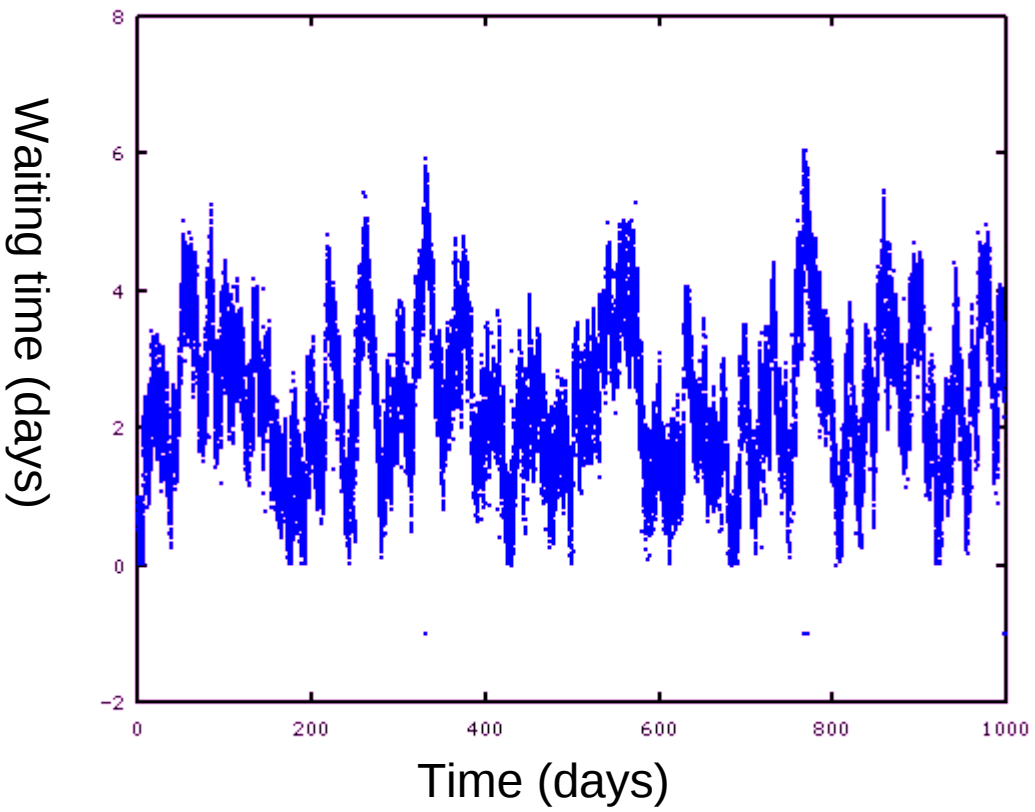

- Waiting time < 1 day
- No canceled tasks
- Average system load is 74 % (e.g. in average 975 CPUs are idle all the time)

Fixed limit system does not allow to achieve optimal system load

# Dynamic Limits: Credit System

- Each user have TotalResources/NumberOfusers credits
- For these credits the user can buy CPU/days
- Price for CPU/day changes in time:
  - Price = $N_{\text{requested CPUs}} / N_{\text{Total CPUs}}$
- Most of the tasks are scalable. The users can shrink the tasks if the price for CPU/day is high and extend the tasks if the price is low.

# Results For Credit System




- Average system load 99.7 %
- Average waiting time ~ 2 days
- Cancellation rate<1% for all user groups

Credit system allows the users to adjust their tasks in time.

This allows to optimize the system load, waiting time and cancellation rate

# Advantages Of The Credit System

- No user can occupy the resource forever (nobody has infinite credits)
- Prices for the resources are flexible and depend on the current system load. This make the system self-adjustable.
- People see what they are buying. Nobody will be disappointed or unsatisfied due to the “wrong resource allocation policy”, large waiting time, high cancellation rate and so on.